Realization of Cutter Radius Compensation Using Embedded System in CNC Machine

Shaoxu Wang*[1], Sannan Yuan[2]

School of Electric and Information Engineering, Shanghai University of Electric Power, Shanghai 200090, China

*[1]sxwang@shiep.edu.cn; [2]Samuel.yuan@shiep.edu.cn

Abstract

Cutter radius compensation (CRC) plays a very important role in the NC system. This article discusses the principle and method of radius compensation. It introduces the details of linear motion control algorithm of left and right cutter compensation according to G code instructions, and the implementation and processing methods at every stage of building up, up process and cancellation of cutter compensation. It is implemented with embedded system. In the end a specific example of implementation is given. Experimental results show that the algorithm is correct, and having higher efficiency.

Keywords

Cutter Radius Compensation; Realization Algorithm; Embedded Systems

Introduction

CNC system tracks use the center of the cutter as the reference point in the CNC machining process, it is the same way when designing processing procedure programs. However, the actual processing cutter has a radius and length, therefore, the outline processing completed is different from the actual trajectory of the target. This difference is caused by the cutter radius and length. Cutter radius and length can be considered in the design and processing procedures, the difference is then used to correct the parameters of the program. But this will cause a lot of troubles. If using a cutter with different radius and length, then even processing a same workpiece machining, the processing program is not the same. In some applications, it hopes the first process is roughing, then finish with precise, thus the coarse and fine machining programs are not the same. It is obviously inappropriate. In order to solve these problems, it usually adopts cutter radius compensation (CRC) in machining processing. CRC includes two aspects, which are length compensation and radius compensation. Using compensation, the same machining program can be used by setting different compensation parameters, regardless of different cutters, roughing or finishing. So it has been widely used. CRC includes length compensation and radius compensation. There are many ways to achieve (Zhao Yajun, Meng Wen, Li Jian, 2007; Chen Xiaohong, GU Qijun, MENG Qingbo, 2009; Li Yanxia, 2007). This article discusses the principle and algorithm of CRC, and the implementation with embedded systems. It focused on the way of linear compensation. Experimental results show that the algorithm is correct with very high processing efficiency.

Realization Principle

When CNC machining center works, machining program is designed to track the actual workpiece contour, but because the cutter has radius, cutter enter path is impossible to completely overlap contour. Its size is the cutter radius deviation. For this reason, CNC system automatically corrects the deviation in accordance with program instructions and radius in the processing. This is called cutter compensation. In accordance with the cutter enter path stays in the left or right side of programmed path (i.e., part contour) forward direction, they are called left CRC and right CRC. It is denoted with G41 and G42 in programs. At the end, G40 must be used to cancel the CRC. CRC is essentially to calculate the trajectory of the cutter enter point according to the procedure trajectory and cutter radius. In compensation processing, two aspects should be considered, that's the basic contours and the corners. The basic outline of the processing is relatively simple, as long as deviating from the original track away from r, where r is the radius of the cutter. If it is linear motion, the cutter enter path after CRC is a straight line which is parallel to the contour straight and with distance r. It is easy to achieve as long as the start and end points are calculated. For circular motion, the cutter enter path after CRC is arc which is concentric with the original circular. It can also be achieved as long as the start point, end point and center coordinates of the arc of the cutter enter path are calculated. It is required to determine the

positive or negative compensation in the concrete calculation according to the direction of movement, the compensation mode and circular clockwise. As shown in Figure 1.

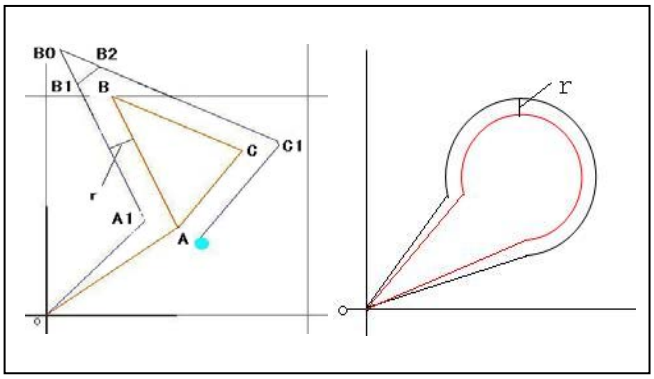

FIG.1A LINEAR MOTION FIG.1B ARC MOTION
FIG.1 SCHEMATIC DIAGRAM OF CRC

For dealing with the corners, the key is the corner point coordinate calculation. The corner point coordinates are involved with cutter radius vector end points and trajectories. For linear motion as shown in Fig.1a, it can be done along with the linear motion which is parallel to and away r from the track of programming. So the actual motion relative to the original path may stretch, or shorten, the corresponding algorithm are called elongation and shortening type. But it can also be seen from A1B0C1 in the chart that the non cutting cutter travel time is bound to increase if still using the elongation type connection. In order to improve the efficiency, the algorithm is modified, which is adding the stroke B1B2, so that the final stroke is A1B1B2C1, this treatment is called insertion type control. The cutter enter path angle type is determined by the angle of front and rear straight line and compensation mode(Ren Yutian, Bao Jie, Yu Yijun, 2004).

TABLE 1 ANGLE TRANSITION TYPE

Cutter compensation Methods	sin θ	cos θ	types
G41	⩾0		shortening
	<0	⩾0	elongation
	<0	<0	insertion
G42	⩾0	⩾0	elongation
	⩾0	<0	insertion
	<0		shortening

Assuming the angle of last program trajectory and the X axis is α, the angle of this program trajectory and the X axis is β, $\theta=\beta-\alpha$, then the switching transition types are shown in table 1.

Calculations of the specific types can be seen in the following CRC algorithm.

Realization Process and Algorithm

If D parameter is input in the G code, then set the cutter radius as the corresponding system configuration cutter radius for D. If input G40, G41 and G42, then set the mode of CRC as Cancel, left CRC, right CRC. For G40 CRC cancel mode, because subsequent G code following the input command should be completely realized according to the CRC method, it should not be immediately cancelled, so it adds a middle stage of cancelling CRC, called "COPM_CANCEL2".Meanwhile, sometimes G code program G40 is followed with G00 or G01 motion command, sometimes G40 is a separate line command, so it should be distinguished to process. If G40 is followed with the motion command, it only sets the CRC mode as COPM_CANCEL, when the motion command is completed, then set to COPM_CANCEL2. If each axis's moving distance is 0 in this line instruction after calculation, G40 is a single line command, then set the CRC mode to COPM_CANCEL2 directly. The final movement instruction processing is executed according to whether the CRC is finished. If it is, ordinary G00 G01 G02 G03 motion commands are executed, otherwise implement CRC motion algorithm.

The movement of CRC instruction algorithm is as follows:

System implementation of G code is based on the behavior of line unit, process once per single line. But the specific track of CRC must obtain by subsequent instruction parameters. So sequence steps should be considered when design algorithm. Meanwhile, because system resources need to apply for each moving command, especially the moving cache memory, the main process needs to judge if resources is adequate before movement is implemented. That is why two moving command cannot be issued at the same time, it must be deployed by main process. So CRC algorithm needs to save a lot of parameters. Here is a detailed description of CRC algorithm with linear motion. When calling this algorithm, the system has calculated the moving destination of linear motion and moving mode of G00 or G01 after analysis of G code. If G01, the speed of feed rate is also got.

Establish of CRC

a.For first input G41 or G42, CRC has not yet been established, CRC flag is 0. Calculate this linear moving destination. Because it should not move directly to the destination, the cutter center position also can't be

drawn. Then record this location as pos[].

b.For convenient calculation, the left and right CRC are comprehensively considered with same formula and different coefficient. If it is left CRC, set coefficient isleft=1, if it is right CRC, set coefficient isleft= -1.

c.Calculate the angle between this line and the X axis as alpha.

d.Initializer[x]=0, r[y]=0,where r[x], r[y] are projection of tool center position of last program segment to starting point O of this program segment in X and Y axis. Record the moving mode of G00 or G01 and moving speed of feed rate;

e.Because the program cannot move now, the moving offset oa[] need to calculate and record.

f.Set CRC flag to 1, system state to CRC cycle;

The CRC algorithm cannot formally issue a move instruction. The real implementation is issued by main loop process. Therefore it needs to set the system state. If it is CRC cycle, the main process issues a moving instruction in accordance with CRC planning.

Process of CRC

Entering program of CRC, if the process has been established, namely CRC flag is 1, then CRC works, the actual is performing the last program segment moving command according to this program segment parameters.

a.Calculate the coordinates of the program segment.

b.Calculate the angle β between the line and the X axis. The angle θ between this line segment and last segment θ=β-α.

c.If sinθ>=0, then: if left CRC, calculate shortening parameters, else if cosθ>=0, then calculate elongation parameters, otherwise calculate insertion parameters, as shown in figure 2.

d.If sinθ<0, then: if right CRC, calculate shortening parameters, else if cosθ>=0, then calculate elongation parameters, otherwise calculate insertion parameters.

e.For shortening or elongation movement, calculate the moving offset sc[],set CRC step to CONNECT_SC, and calculate the next iteration of r[x], r[y]. For inserting movement, first calculate moving offset sc[], then calculate interpolation moving offset cc[],set CRC step to CONNECT_SCCC, and calculate the next iteration of r[x], r[y].

f.In order for the next iteration, recording parameters:β substitutes α, moving offset of this segment oa[]= endpoint position of the program segment - point position pos[] of last program segment. Save endpoint location assignment in pos[]. Save movement way G00 or G01 of this segment, and the moving speed of feed rate.

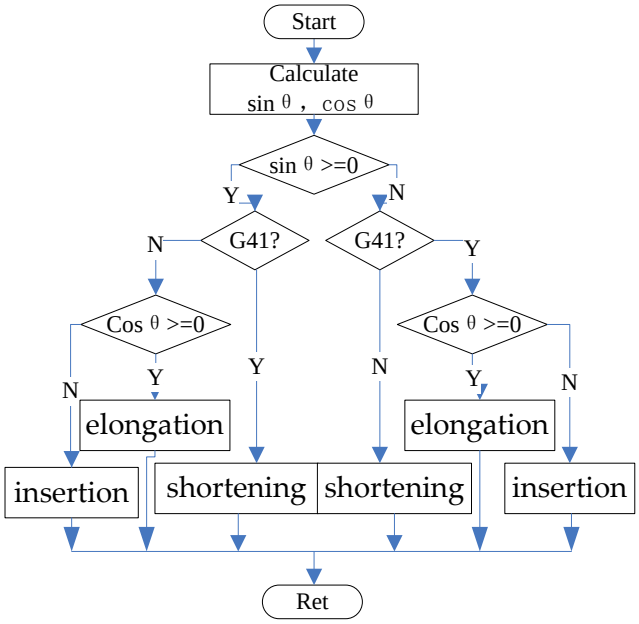

FIG.2 CONNECTING TRANSITION ALGORITHM

For the elongation and shortening type, the last segment of the actual moving distance is shown in formula(1):

$$sc[x] = oa[x] + ac[x] - r[x]$$
$$sc[y] = oa[y] + ac[y] - r[y] \quad (1)$$

Where oa[x], oa[y] is moving distance of this segment, ac[x], ac[y] is shown in formula 2.

If G41, en

$$ac[x] = -r * (\sin\alpha + \sin\beta)/ (1+\cos\theta)$$
$$ac[y] = r* (\cos\alpha + \cos\beta)/ (1+\cos\theta)$$

If G42, en

$$ac[x] = r * (\sin\alpha + \sin\beta)/ (1+\cos\theta)$$
$$ac[y] = -r* (\cos\alpha + \cos\beta)/ (1+\cos\theta) \quad (2)$$

Then ac[x], ac[y] are assigned to r[x], r[y] of the next iteration.

For inserting type, the actual movement of last program code is divided into two segments, the relative moving distance are shown as formula (3):

$$sc1[x] = oa[x] + ac1[x] - r[x]$$
$$sc1[y] = oa[y] + ac1[y] - r[y]$$
$$sc2[x] = ac2[x] - ac1[x]$$
$$sc2[y] = ac2[y] - ac1[y] \quad (3)$$

where oa[x],oa[y] are relative moving distance of the program segment,ac1[x], ac1[y], ac2[x], ac2[y] are shown as formula (4):

If G41, then

$$\begin{aligned} ac1[x] &= -r * (\sin\alpha - \cos\alpha) \\ ac1[y] &= r * (sina1 + \cos\alpha) \\ ac2[x] &= -r * (\sin\beta + \cos\beta) \\ ac2[y] &= -r * (\sin\beta - \cos\beta); \end{aligned}$$

If G42, then

$$\begin{aligned} ac1[x] &= r * (\sin\alpha + \cos\alpha); \\ ac1[y] &= r * (sina1 - \cos\alpha); \\ ac2[x] &= r * (\sin\beta - \cos\beta); \\ ac2[y] &= r * (\sin\beta + \cos\beta); \end{aligned} \quad (4)$$

Then ac2[x], ac2[y] are assigned to r[x], r[y] of the next iteration.

Cancellation of CRC

If program instruction is G40, it means CRC cancellation. Before the abolition instruction movement, it needs to complete the last unfinished movement. Therefore, it is treated as follows: If received G40 with G00 and G01 motion command, set CRC mode as COPM_CANCEL, process this segment, then set the mode as COPM_CANCEL2. If G40 is a separate instruction, then directly set mode as COPM_CANCEL2.

a.If CRC mode is not COPM_CANCEL2, then process it in accordance with the normal CRC.

b.Otherwise calculate the offset of last segment, save as sc[]. Then calculate the offset of arriving at target, save as cc[]. Set CRC step as CONNECT_SCCC. Set CRC mode as COPM_OFF. Clear CRC flag.

In main process, if system state is CRC cycle, then process CRC according to CRC steps, otherwise perform normal non CRC processes.

a.If CRC step is CONNECT_SCCC, perform a moving with offset sc[], then change step to CONNECT_CC.

b.If step is CONNECT_SC, perform a moving with offset sc[],change step to CONNECT_RESET.

c.If step is CONNECT_CC, perform a moving with offset cc[], change step to CONNECT_RESET.

d.If there is no new instruction after performing a movement, then change CRC mode to COPM_CANCEL2 if it is COPM_CANCEL. Otherwise set system cycle status to normal mode if CRC mode is COPM_OFF.

Fig.3 shows a simple example of CRC, completing a triangular cutting, using left CRC. According to the above algorithm, the result is shown in Fig.3B.

```
N010g0x0y0z0
N020g41g0x10y8d1
N030g1x5y20f2000
N040x15y15
N050g40x10y8
N060g0x0y0
```

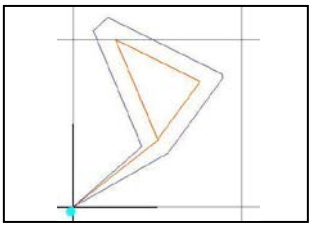

FIG.3A G CODE PROGRAM
FIG.3B COMPLETE MACHINING PROCESS
FIG.3 EXAMPLE OF CRC

Conclusions

In this paper, CRC algorithm of NC system is studied. It is realized using embedded system. The embedded controller is based on ARM Cortex M4.The results show that the algorithm is correct, it realizes the function of CRC, and its efficiency is very high. The system has good application prospects.

REFERENCES

CHEN Xiaohong, GU Qijun, MENG Qingbo, Research on function of cutting tool compensation [J], Modern Manufacturing Engineering, 2009(3).

Li Yanxia, Study of application on cutter radius offset of numerical control milling [J], Modern Manufacturing Engineering, 2007(8).

Ren Yutian, Bao Jie, Yu Yijun. New CNC machine tool technology [M], Beijing: Beijing Institute of Technology Press, 2004.

Wang Yang. CNC machine spline cutter radius compensation algorithm research [J], Mechanical Design and Manufacturing, 2007(6).

Zhao Yajun, Meng Wen, Li Jian. Cutter radius compensation in CNC Simulation [J], Mechanical Research & Application, 2007(06) .

Zhao Yajun, Meng Wen, Li Jian. Application of tool radius compensation in simulation of numerical control [J], Mechanical Research & Application, 2007(6).